Assignment Guidance and Front Sheet

This front sheet for assignments is designed to contain the brief, the submission instructions, and the actual student submission for any WMG assignment. As a result the sheet is completed by several people over time, and is therefore split up into sections explaining who completes what information and when. Yellow highlighted text indicates examples or further explanation of what is requested, and the highlight and instructions should be removed as you populate 'your' section.

This sheet is only to be used for components of assessment worth more than 3 CATS (e.g. for a 15 credit module, weighted more than 20%; or for a 10 credit module, weighted more than 30%).

To be completed by the student(s) prior to final submission:

Your actual submission should be written at the end of this cover sheet file, or attached with the cover sheet at the front if drafted in a separate file, program or application.

Student ID or IDs for group work	**U2136249**

To be completed (highlighted parts only) by the programme administration after approval and prior to issuing of the assessment; to be consulted by the student(s) so that you know how and when to submit:

Date set	24/02/2023
Submission date (excluding extensions)	28/04/2023
Submission guidance	Submit to Tabula
Marks return date (excluding extensions)	26/05/2023
Late submission policy	If work is submitted late, penalties will be applied at the rate of **5 marks per University working day** after the due date, up to a **maximum of 10 working days** late. After this period the mark for the work will be reduced to 0 (which is the maximum penalty). "Late" means **after the submission deadline time as well as the date** – work submitted after the given time even on the same day is counted as 1 day late. For **Postgraduate** students only, who started their **current course before 1 August 2019**, the daily penalty is **3 marks** rather than 5.
Resubmission policy	If you fail this assignment or module, please be aware that the University allows students to remedy such failure (within certain limits). Decisions to authorise such resubmissions are made by Exam Boards. Normally these will be issued at specific times of the year, depending on your programme of

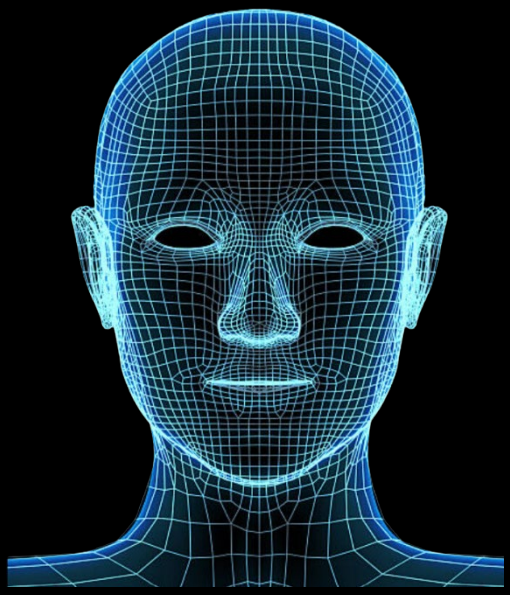
	study. More information can be found from your programme office if you are concerned.

To be <u>completed</u> by the <u>module owner/tutor</u> prior to approval and issuing of the assessment; to be <u>consulted</u> by the <u>student(s)</u> so that you understand the assignment brief, its context within the module, and any specific criteria and advice from the tutor:

Module title & code	WM245 Programming Languages for Cyber Security
Module owner	
Module tutor	Dr Nikki Williams
Assessment type	Cyber Tool Report
Weighting of mark	60%

Assessment brief
Please see below

Word count	Described below
Module learning outcomes (numbered)	1. Compare different programming paradigms used to create software. 2. Reflect on how software vulnerabilities can be minimised during software creation. 3. Incorporate security features in small-scale programs. 4. Develop small-scale programs that employ the idioms of a programming paradigm in a conventional manner.
Learning outcomes assessed in this assessment (numbered)	1. Compare different programming paradigms used to create software. 3. Incorporate security features in small-scale programs. 4. Develop small-scale programs that employ the idioms of a programming paradigm in a conventional manner.
Marking guidelines	Generally indicated within specification
Academic guidance resources	All queries to be directed to the tutor's email, with responses posted via moodle or workshop sessions.



FaceSecure

A Facial Biometric Authentication Tool - Report

Contents

Introduction

As cyber threats continue to increase (AAG, 2023), individuals and organisations face a major challenge in securing their information. To combat the rise of cyber attacks and data breaches, stronger authentication methods are needed to protect sensitive data (NCSC, 2023). "FaceSecure," a facial biometric authentication tool, is designed to provide a more secure and efficient method of authentication using live face detection and recognition.

FaceSecure is suitable for a range of industries, including financial service providers, healthcare providers, and government agencies, where protecting sensitive information is critical. It leverages the power of OpenCV; an open-source computer vision library, to provide accurate, real-time image processing capabilities (OpenCV, 2019). With a variety of facial recognition algorithms, FaceSecure can quickly and accurately authenticate users based on their biometric data, ensuring that only authorised personnel can access sensitive information.

Biometric Decision Making and Facial Recognition Algorithms

I started the development of FaceSecure by deciding on which biometric authentication mechanism to use for the verification of individuals. Biometrics are “the automated recognition of individuals based on their biological and behavioural characteristics” (ISO and IEC, 2023). There are numerous experimental biometrics which measure such things as gait, heartbeat and skin reflectance, but to meet the requirements I only focused on those which are commercially available and appropriate to the field of cyber security, including; fingerprint, face, pattern, iris and speaker recognition.

Biometrics work differently than a PIN or password. In these cases, an access control system will compare a stored password with the one entered by an individual, if they are identical, access will be granted. However, no two captures of biometric data will produce a truly ‘identical’ results (EDPS, 2020). So, a biometric system such make an estimation as to whether two biometric samples come from the same individual (NCSC, 2019).

I decided to use face recognition as a means of login authentication as it is simple to integrate with security software, faster and more accurate way to identify individuals compared to other biometric techniques like fingerprints or using SMS notification as two-factor authentication (Amazon, 2021).

When choosing a facial recognition algorithm, I opted to use the Local Binary Pattern (LBP) algorithm fur to its simplicity to implement, speed and accuracy of its results, to ensure that the only the authorised user can access the client’s sensitive information. An alternative option for authentication would have been to use a time-based token system. In this system, users would be required to enter a time-based token in addition to their username and password. However, the use of real-time authentication was chosen as it provides a simpler and more efficient authentication method for users.

The LBP is a texture descriptor algorithm that extracts features from an image. It works by comparing each pixel’s intensity values with its surrounding pixels and encoding the results into a binary pattern. The patten is then used to describe the texture of the image (Ahonen, Hadid and Pietikäinen, 2004). The main reason why I chose the LBP algorithm was because it is simple and fast so it can used to perform real-time facial recognition on low-powered devices. This makes it ideal for use in FaceSecure, where clients need to authenticate users in real-time without experiencing any significant delay. An additional factor that contributed to my decision to use the LBP algorithm was its proven high-level accuracy in facial recognition tasks, even when dealing with challenging scenarios like changes in lighting conditions, facial expressions and occlusions (Shankar and Udupi, 2016). Although, LBP has some limitations when dealing with changes in facial hair significant changes in appearance due to aging , in the context of FaceSecure, where the user’s facial biometric data is likely to remain constant, this was not a significant concern (Marr, 2019).

For the purposes of this tools, I decided to use express.js framework – a robust web application framework – to utilise JavaScript and Node.js as the programming language for developing FaceSecure. This framework provided me with many server-side features such as middleware support, routing and handling HTTP requests and responses (nodejs, 2023). I chose this stack as it is lightweight, fast and provides excellent performance. I utilised OpenCV as the main computer vison library as its designed for real-time computer vision and supports model execution for Machine Learning (ML) essential for FaceSecure to train the algorithm models for real-time vision tasks, such as face detection and facial recognition training under different lighting conditions or from different angles. OpenCV is vastly used and supported by a large community, which ensures that it will be regularly updates with more features and security patches (OpenCV, 2020). An alternative was to use a commercial face recognition API such as Amazon Rekognition or Microsoft Azure Face API. However, I decided against this option because it would have meant that user's facial data would be sent to third-party server, which could compromise the privacy and security requirements of client users of governmental agencies or financial institutions that utilise FaceSecure as means for secure authentication.

FaceSecure Development Process and System Architecture

To create FaceSecure, I adopted the agile methodology approach and divided the program into specific sprints. I identified the following steps as the optimal way to develop the tool:

1. Develop a facial detection and recognition method (tested with a minimum of two subjects)
2. Create front-end webpages for each login phase of the user.
3. Develop a server to deliver the webpages to the user during specific events.
4. Authenticate a user with a username and password (obtained from the initial webpage)
5. Capture an image with webcam and transmit it to the server for processing.
6. Implement the newly captured image in the facial recognition method to authenticate the user.
7. Deliver final webpage to authorise the login of a user after passing the two-factor facial authentication.

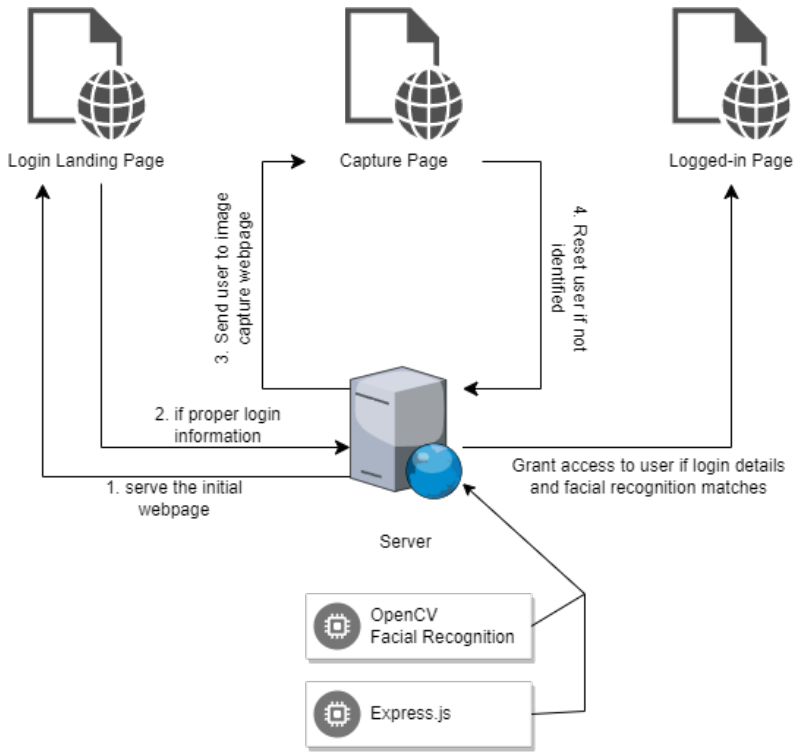


Figure 1 - FaceSecure Functionality Architecture

The tool's functional architecture is presented in Figure 1. To receive content from user form action on the webpages, I utilised Express.js's POST and GET method algorithms. Each method prepared an input in a JSON format and verified it against known information, such as a valid username/password combination or whether the user's image contains a recognised subject. In the initial login authentication phase, the user is presented with an HTML file containing two field inputs, username and password. After entering the information, the server verifies it against its records, and if valid, the camera input file is provided to the user. If not, the user is required to authenticate again.

Once the user takes a photo, JavaScript on the front-end captures a snapshot of the canvas and converts it to an image, storing the base64 encoding on the image. The server then converts it back to PNG format and stores it in the server's directory of faces to use during the face recognition phase. The face recognition method is then called, which involves converting all images to grayscale, detecting faces within images, mapping faces to names, and applying the LBP algorithm to verify if the user is a known subject to finally grant system access.

To ensure the security of user data, error handling and input validation were implemented throughout the development process. One important aspect of FaceSecure is the detection and handling of errors that may occur during the face recognition process. For example, if the user's face is not detected or recognised properly, the program will display a graceful error message prompting the user to try again. This security feature prevents attackers from exploiting weaknesses in the face recognition system by potentially providing false data or attempting to bypass the authentication process.

Spoof attacks are a common type of cyber threat in this context where a malicious actor uses fake or artificial representation of a real object texture to gain unauthorised access to perform malicious actions. In the case of FaceSecure, spoof attacks can occur when an attacker ties to bypass the facial biometric authentication system by using a fake subject face or a still image of a real face.

To prevent this, several anti-spoofing measures have been implemented. One such measure is liveness detection, which ensures that the face detected is alive and not just static image. Liveness detection involves checking for specific facial movements such as eye blinking, head nodding, or mouth movements, which are difficult to replicate using a still image or a fake face. The system has been designed and trained to detect printed, digital images and 3D masks, which are common types of spoof attacks, it utilises existing library methods to analyse the texture of the face, to distinguish between a real and fake face.

Conclusion

In conclusion, the development of FaceSecure demonstrates the importance of considering security and privacy concerns throughout an entire development process. By utilising the latest advancements in facial recognition technology and leveraging OpenCV's computer vision library for live face detection and recognition, I have designed a secured and user-friendly authentication tool that provides a secure and robust solution for individuals and organisations looking to enhance their cyber security infrastructure, facilitating a higher level of security than traditional methods. The choices made during the development process, including the programming languages and libraries were carefully considered and aimed at ensuring the overall security of the system. However, as new security threats emerger, it will be important to remain vigilant and continue to update and enhance the security of the system by implementing more security features including advanced algorithms such as Bcrypt to protect the integrity of biometric data by converting data of any size into a fixed size string of characters, ensuring that even if an attacker gains access to biometric data, they will not be able to determine the original image used for authentication (Council, Pato and Millett, 2020).

References

AAG (2023). *The Latest Cyber Crime Statistics (updated March 2023) | AAG IT Support*. [online] aag-it.com. Available at: https://aag-it.com/the-latest-cyber-crime-statistics/#:~:text=Cyber%20Crime%20Overview.

Ahonen, T., Hadid, A. and Pietikäinen, M. (2004). Face Recognition with Local Binary Patterns. *Lecture Notes in Computer Science*, pp.469–481. doi:https://doi.org/10.1007/978-3-540-24670-1_36.

Amazon (2021). *What is Facial Recognition - Beginner's Guide to Face Analyzer Software and Machine Learning - AWS*. [online] Amazon Web Services, Inc. Available at: https://aws.amazon.com/what-is/facial-recognition/#:~:text=Facial%20recognition%20is%20a%20quick.

Council, R., Pato, J.N. and Millett, L.I. (2020). *Introduction and Fundamental Concepts*. [online] Nih.gov. Available at: https://www.ncbi.nlm.nih.gov/books/NBK219892/.

EDPS (2020). *Misunderstanding with regard to biometric data*. [online] Available at: https://edps.europa.eu/sites/edp/files/publication/joint_paper_14_misunderstandings_with_regard_to_identification_and_auuthentication_en.pdf.

ISO and IEC (2023). *Information Security, Cybersecurity, and Privacy Protection*. [online] Iso.org. Available at: https://www.iso.org/obp/ui/#iso:std:iso-iec:24745:ed-2:v1:en [Accessed 1 May 2023].

Marr, B. (2019). *Facial Recognition Technology: Here Are The Important Pros And Cons*. [online] Forbes. Available at: https://www.forbes.com/sites/bernardmarr/2019/08/19/facial-recognition-technology-here-are-the-important-pros-and-cons/?sh=820efb014d16 [Accessed 1 May 2023].

NCSC (2019). *Choosing biometrics*. [online] www.ncsc.gov.uk. Available at: https://www.ncsc.gov.uk/collection/biometrics/choosing-biometrics.

NCSC (2023). *Current challenges*. [online] www.ncsc.gov.uk. Available at: https://www.ncsc.gov.uk/section/ncsc-for-startups/current-challenges.

nodejs (2023). *opencv4nodejs*. [online] npm. Available at: https://www.npmjs.com/package/opencv4nodejs.

OpenCV (2019). *OpenCV library*. [online] Opencv.org. Available at: https://opencv.org/.

OpenCV (2020). *Face Recognition with OpenCV — OpenCV 2.4.13.7 documentation*. [online] docs.opencv.org. Available at: https://docs.opencv.org/2.4/modules/contrib/doc/facerec/facerec_tutorial.html.

Shankar, S. and Udupi, V.R. (2016). Recognition of Faces – An Optimized Algorithmic Chain. *Procedia Computer Science*, 89, pp.597–606. doi:https://doi.org/10.1016/j.procs.2016.06.020.